REMARKS

Reconsideration of the above-identified application, as amended, is respectfully requested.

In the Official Action dated March 30, 2004, the Examiner first objected to the Claims 1, 3, 4 and 8 as comprising minor informalities. Applicants' take this opportunity to correct the errors indicated by the Examiner in each of Claims 1, 3, 4 and 8.

The Examiner further rejected Claims 1-4, 6-8, 15-19, 21-23 and 28 under 35 U.S.C. §103(a) as allegedly being unpatentable over Relph (U.S. Patent No. 6,092,171) in view of Moore (U.S. Patent No. 6,564,305). The Examiner further objected to Claims 5, 9-14, 20 and 24-27 as being dependent upon a rejected base claim, but indicated that these claims would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

With respect to the rejection of independent Claims 1 and 16, Applicants respectfully disagree in view of the amendments and arguments in traversal herewith.

Particularly, a key feature of the present invention is the fact that the method for managing real memory usage in the present invention and the compressed main memory control device for managing real memory usage operates transparent to the operating system kernel, i.e., without modification to the operating system. This, it is respectfully submitted, is neither taught nor suggested by Relph or Moore references, whether taken alone or in combination.

Relph, for instance, deals with a compressed memory system in a peripheral device, i.e., a laser printer (including an embedded controller), and particularly, to a method for storing data in the peripheral from a host computer in storage unit subsets. In response to reaching a threshold storage level, Relph teaches generating a signal and compressing data in said

subsets when the threshold storage is reached. This enables further data to be stored in uncompressed form and, the process of compressing data and moving to make room for uncompressed data storage in the peripheral is continuous. The system described in Relph, however, is such that the Operating System (O/S) is aware of the interaction with the compressed memory device, i.e., the management of data in peripheral memory is effected by the O/S kernel.

While the Examiner has correctly indicated that Relph does not teach memory usage in a computer system that is transparent to the operating system, he cites Moore at Col. 1, lines 46-50 as allegedly teaching operating system transparency, i.e., memory compression in a computer device wherein memory usage is transparent to the O/S. Applicants respectfully disagree.

Moore, like Relph, is associated with a specific device that is a combination of a hardware device and a control program written for that specific hardware device which includes compressed memory. In Moore, the page manager is aware of compression and gets involved. Though it may be transparent to *applications* running on that device (e.g., a laser printer), like Relph, it is not transparent to the control program or O/S. That is, in the prior art device of Moore, the control program and O/S is modified to become aware that there is compressed memory and thus includes functions to interact with and effect control of memory management. While the Examiner has cited taught Moore at Col. 1, lines 46-50 as teaching operating system transparency, applicants respectfully submit that the cited passage suggests implementation of mechanisms for transferring information between compressed and uncompressed domains in a manner that is *transparent to applications and other software elements executing* (See Moore at Col. 1, lines 46-52, emphasis added). Thus, respectfully, the Examiner's misunderstanding is

misplaced because in Moore at Col. 3, lines 36-52, Moore specifically talks about memory control being wholly within the domain of the O/S.

Unlike the prior art, the present invention is directed to a computer system having a new function of compressed memory and an off-the-shelf O/S, e.g., O/S like Microsoft ("MS") Windows. It is critical to the understanding of this invention that only Microsoft can change its operating system and is not sold as part of the computing device, and, thus, is a separate piece of software, not written for the specific compression feature of the invention. The invention comprises non-kernal program (without modification to O/S or external thereto), i.e., software, which runs independent of the O/S and is a completely separate system for handling compressed memory management in a computing device.

Further, while the Examiner alleges that Moore teaches compression management, the present invention does not teach implementation of compression, but management of memory in a compressed system.

Thus, both prior art references teach modification of operating systems and each O/S in Relph and Moore is completely aware that there is a compressed memory device in the system. On the other hand, in the present invention, the O/S has no idea that there is compressed memory in the system. That is, in the present invention, software runs independently of the O/S while using mechanisms (functions in the O/S) in the O/S for assisting in management of compressed memory. In the present invention, there is no logic in the O/S that indicates the use of compressed memory system much less knowledge of the current state of the compressed memory system, i.e., compression ratio. Thus, the present invention is a software program that is not written into the O/S, i.e., operates independent of and transparent to the control system or operating system, but is designed to act in concert with the generic O/S. Independent Claims 1

and 16 have been amended to further clarify that the control of the real memory usage in the computer system is accomplished without modification to an operating system kernel and transparent to the operating system.

Respectfully, no new matter is being entered by this amendment as the specification and drawings (e.g., Figure 1) as originally filed describe the inventive features (i.e., compression management service) outside the O/S kernel subsystems 75.

In further support for patentability of the claims, it is submitted that even if the system of the invention was integrated into the operating system, as in the prior art, it would not perform in the same way, i.e., by freeing memory and initiating allocation/recovery. That is, for example, the present invention incorporates "memory eater" processes (e.g., as set forth in dependent Claims 9 and 24) that perform allocate/recover memory to/from the within the operating system's memory space. This concept is neither taught nor described in the combination of prior art references cited.

A further amendment is being made to the preambles of Claims 1 and 16 to clarify that that the real memory is characterized as a fixed amount of main memory as seen by a processor. Respectfully, no new matter is being entered by this amendment as full support for this limitation is clear from page 4, lines 2-3.

Further with respect to the rejection of Claims 2 and 17, the Examiner indicates as the basis for the rejection that Relph teaches the generation of a signal to determine when to compress/decompress the content of memory (via the operating system). The invention rather teaches generation of an interrupt to determine when to move content out of memory to a swap device which is a completely different mechanism. To clarify the concept of the invention, memory is allocated to drive the operating system to reduce its usage of certain classes of

memory, for example, which results naturally in swapping, but may result in other types of controlled reduction of memory usage, e.g., trimming the size of the cache, etc. It is stressed that the driving of the O/S (in the invention) is not through any application API, but underneath the O/S that causes its natural mechanisms to implement the desired controls.

If the O/S is modified to support compressed memory, as in the prior art Relph and Moore references, the virtual memory manager would have to include logic to become aware of physical memory, virtual memory in addition to a third dimension of memory, i.e., compressed memory. The present invention does not modify O/S; the present invention rather has taken all of this logic outside the O/S to manage compressed memory using facilities of the O/S.

Further amendments are being made herein to Claims 8 and 23 to clarify that the compressed memory manager system sets the thresholds based on statistics of memory usage. Respectfully, no new matter is being entered in the claims as full support for the controller chip (memory) to set the threshold to generate an interrupt. That is, a user (a human operator) is not setting a threshold, the software has been programmed to do so.

As the invention further provides the mechanism of how to improve the compression ratio to improve by placing trivial data pattern, e.g., all zeros in virtual memory space eaten by the memory eaters, new claims 29-30 are being added that are directed to this feature. That is, new Claims 29-30 are directed to the specific memory eater process that functions to take away memory from the system by replacing the data in virtual pages of memory with a highly compressible data, e.g., a trivial data pattern of all zero's. Thus, the average compressibility of all real memory is controllable by artificially populating it with larger/smaller tracks of compressible data ("zero page ops"). This feature is described in the specification at page 11, first full paragraph, for instance, where it is described how the memory eater process

adds zero bit patterns to manage average compressibility of all data stored in memory by populating (filling) pieces of a fixed virtual address space with highly compressible data. Respectfully, no new matter is being added.

This is a patentable feature over the prior art where selective pieces of data are moved out of memory. Rather, in the present invention, the compression ratio is monitored, and when it turns unmanageable, large tracts of memory are artificially populated with highly compressible data. As the O/S will know how to keep its total amount of address space it has in memory under control, then as previously undefined virtual pages are allocated with highly compressible data, it has to move other data out of its virtual address space to make room. The O/S will know how to move this data out of the memory. Thus, the present invention is able to artificially depopulate uncompressible data in memory and replaces it with highly compressible data.

In the analysis of the present invention, for the type of hardware contemplated in the invention, e.g., a general purpose computing device such as a personal computer (e.g., IBM PC), the implementation of the O/S (e.g., Microsoft O/S) can not be modified, therefore, for MS software to run on IBM hardware, must be done without requiring the O/S to be changed. This is contrary to the exemplary scenario in the case of a Linux operating system, where the O/S kernel is open and the memory manager may be changed in the kernel which is a preferred way of tackling such problems.

In view of the foregoing remarks herein, it is respectfully submitted that this application is in condition for allowance. Accordingly, it is respectfully requested that this application be allowed and a Notice of Allowance be issued. If the Examiner believes that a

telephone conference with the Applicants' attorneys would be advantageous to the disposition of this case, the Examiner is requested to telephone the undersigned.

Respectfully submitted,

Steven Fischman
Registration No. 34,594

SCULLY, SCOTT, MURPHY & PRESSER
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343
SF:gc